Listing of Claims

This listing of claims replaces all prior versions, and listings, of claims in the application:

1. (Previously Presented) A method comprising:

receiving a plurality of instructions from a test interface;

loading the plurality of instructions into an emulation instruction register;

receiving a plurality of instructions from the emulation instruction register;

determining a validity of a first instruction of the plurality of instructions by reading width bits in the first emulation instruction;

providing the first instruction to a decoder of the processor if the first instruction is valid;

without receiving a run-test idle state signal, determining a validity of a second instruction of the plurality of instructions by reading width bits in the second instruction; and

providing the second instruction to the decoder if the second instruction is valid.

2. (Canceled)

3. (Previously Presented) The method of Claim 1, further comprising storing the plurality of instructions in the emulation instruction register in subsequent clock cycles.

4. (Canceled)

5. (Previously Presented) The method of Claim 1, further comprising loading the plurality of instructions in parallel into the emulation instruction register.

6. (Previously Presented) The method of Claim 1, further comprising providing the second instruction to the decoder after the first instruction is completed.

7. (Previously Presented) The method of Claim 1, further comprising providing the plurality of instructions to the decoder after a first run-test idle state without entering into a second run-test idle state.

8. (Previously Presented) The method of Claim 1, further comprising providing the first and second instructions to a digital signal processor.

9. (Currently Amended) A method of providing instructions to a processor, the method comprising:

loading a plurality of instructions into an emulation instruction register;

receiving a run-test idle state signal;

providing the plurality of instructions to the processor <u>in response to the receipt of the run-test idle state signal</u>; and

processing the plurality of instructions without receiving another run-test idle state signal.

10. (Canceled)

11. (Previously Presented) The method of Claim 9, further comprising determining a validity of each of the plurality of instructions before processing by reading bits in each instruction indicating a width of the instruction.

12. (Previously Presented) The method of Claim 11, further comprising aborting processing of any invalid instructions and loading a next instruction of the plurality of instructions from the instruction register.

13. (Previously Presented) The method of Claim 9, further comprising loading a next instruction of the plurality of instructions from the instruction register if a no-operation instruction is loaded.

14. (Previously Presented) The method of Claim 9, further comprising providing the plurality of instructions to the processor a plurality of times without reloading the instruction register.

15. (Original) The method of Claim 9, further comprising providing the plurality of instructions to a digital signal processor.

16. (Previously Presented) A processor comprising:

a test interface;

an emulation instruction register adapted to store a plurality of emulation instructions received from the test interface;

emulation control logic adapted to control a flow of the plurality of emulation instructions to a processor pipeline following detection of a single run-test idle state; and

a decoder to receive the plurality of instructions for processing.

17. (Canceled)

18. (Previously Presented) The processor of Claim 16, wherein the emulation control logic determines a validity of the plurality of instructions by reading bits in each instruction indicating a width of each instruction and discards any invalid instructions.

19. (Previously Presented) The processor of Claim 16, wherein the emulation control logic loads a next instruction from the emulation instruction register immediately after detecting a no-operation instruction.

20. (Original) The processor of Claim 16, wherein the processor is a digital signal processor.

21. (Currently Amended) An apparatus, including operating instructions residing on a machine-readable storage medium, for use in a device to handle a plurality of emulation instructions, the operating instructions causing the device to:

load the plurality of emulation instructions into a single emulation instruction register;

enter a run-test idle state;

provide the plurality of emulation instructions to a processor <u>in response to entry into the run-test idle state</u>; and

process the plurality of emulation instructions.

22. (Canceled)

23. (Previously Presented) The apparatus of Claim 21, wherein a validity of each of the plurality of instructions is determined before processing by reading bits in each instruction indicating a width of each instruction.

24. (Previously Presented) The method of Claim 1, further comprising:

scanning instructions from an in-circuit emulator (ICE) to the test interface, the test interface comprising a Joint Test Action Group (JTAG) interface.

25. (Previously Presented) The method of Claim 1, wherein a pre-determined set of width bits indicates an instruction is invalid.

26. (Previously Presented) The processor of Claim 16, wherein the emulation instruction register comprises first and second registers.

27. (Previously Presented) The processor of Claim 16, wherein the emulation control logic comprises a state machine.

28. (Previously Presented) The processor of Claim 16, further comprises a multiplexer to select between an instruction for the plurality of instructions to send to the processor pipeline.

29. (Previously presented) The apparatus of Claim 21, further comprising an in-circuit emulator to monitor operations of the processor.

30. (Previously Presented) The method of Claim 1, further comprising executing at least one of the plurality of instructions to monitor operation of the processor.

31. (Previously Presented) The method of claim 1, further comprising:

performing a debugging operation using the first and second instructions.